Contour-based cane extraction for 2D vine modelling

November 11, 2013

Jared Klopper

jjk52@uclive.ac.nz

Department of Computer Science and Software Engineering

University of Canterbury, Christchurch, New Zealand

Supervisor: Richard Green

richard.green@canterbury.ac.nz

Abstract

Modelling grape-vines from a two-dimensional image has a number of interesting problems which have to be overcome. Due to their growth patterns and the surrounding environment vines tend to grow in ways which result in a large number of occlusions from a single perspective, making identification difficult. In addition the presence of wires, posts and other foreign objects increase complexity. We propose a method which uses the contours extracted from an image to identify canes. An emphasis has been made on identifying only sections of canes which are not occluded. Our method extracts the contours, creates a model of all edges and pairs up edges which contain similar properties such as their width and direction. High quality edge pairs are then selected to produce partial cane models. These models have been quantitatively assessed by comparing their structure to near perfect hand drawn versions. From this analysis we have found a mean overall accuracy of 71.3% ($\sigma = 5.9\%$) for all images used in our study. These results are promising as the models produced have a high correspondence with the ground truth, although are missing some information. Finally we propose several techniques which could be investigated in the future in order to further improve the accuracy and reliability of our method.

Acknowledgements

Richard Green and Tom Botterill for their constant support and help throughout the year on this project.

Contents

1 Introduction

The automated robot pruning project aims to develop a system capable of pruning vines in commercial vineyards without human aid through the use of computer vision [3]. The project makes use of several camera's to detect features such as canes, posts, wires and other debris within a single image. Prior work has been done on detecting canes from these images, however they have not been suitable for practical use. As a result simulated vines are currently being used within the project [3]. In order to begin reconstructing a 3D model of the plant, canes need to be detected within individual frames. It is however not necessary to detect every single cane within a frame, as missing information is preferable to errors in this instance.

Previous work has been done in extracting the structure of vines from images. A variety of skeletonisation methods have been analysed, however none of them produced skeletons which could be directly used for reconstructing the 3D model [11].

Recently additional research has been conducted into the use of expectation maximization for modelling vines [18, 17]. This approach has showcased promising results, however there is opportunity for improving upon this. Run-time performance of the split-and-merge implementation is not real-time, however effective initialisation of the algorithm may speed it up sufficiently. Any pre-processing steps for producing initial models has to be efficient such that the resulting overhead is minimal. Additionally the models produced for initialisation need to be as error-free as possible in order to reduce the amount of work required and increase overall accuracy.

In this paper we propose a technique for automatically extracting the structure of vines from 2D images using a simple rule based system. Our approach makes use of edge information from a background subtracted image of the plants. These edges are then iteratively joined using a number of simple rules, forming fully associated edges of canes. Cane edges are paired up and a model is produced. A quantitative analysis of these models is performed, and several enhancements which could be made in the future to further improve these results are suggested.

1.1 Report Outline

This report begins with an overview of recent work related to this research. We cover our motivation and the overall importance of this research, including previous research done on skeletonisation, expectation maximization, segmentation and general plant and tree reconstruction techniques. Chapter 3 outlines our approach and implementation for reconstructing vines. Chapter 4 contains information about how we have assessed our algorithm and presents the final results. Chapter 5 contains a brief discussion of these results and their relative comparison to skeletonisation methods. Chapter 6 outlines possible future work which could be conducted to improve our method. Chapter 7 presents our final conclusion.

2 Related work

This chapter contains a review of literature and other related work that has been consulted while conducting this research. We cover the motivation behind the research, including the project it is involved with and other related work. In addition we cover skeletonisation algorithms and expectation maximization to extract the structure of vines. Finally we cover other related work in the area of reconstructing plants and tree models from 2D images.

2.1 Motivation

This research is a small part of the automated vine pruning project currently being developed. The project aims to produce a robot which is capable of automatically pruning vines [3]. In order to accomplish this several components are being developed. The robot has to be capable of constructing a full 3D model of the vines and the surrounding environment in order to be able to make pruning decisions and navigate effectively. The vision side consists of detecting features within 2D images captured from the 3 optical cameras on the rig. Within these 2D frames, features of interest such as canes, posts, wires and vineyard debris are detected. These features are then tracked between subsequent frames in order to build up a 3D model of the environment. Using this 3D model decisions can be made on how to effectively prune the plant.

In order to reliably reconstruct models of canes in varying environments the robot is equipped with a full canopy and lighting setup. This reduces the effects of lighting which is often an issue in vision related work [25]. Additionally laser lines are present which allows depth information to be calculated [3, 2]. The system is currently capable of detecting posts and wires from the 2D images [3]. Additionally work has been done to produce a system capable of making pruning decisions [7] which has shown promising results.

Currently the project is making use of simulated vines when constructing the 3D model [3]. This places an emphasis on reliably extracting the vines from 2D images in order for the rest of the project to make further progress. Previous work has been undertaken in order to extract the structure of the vines using skeletonisation [11], however this was found to be inadequate for use in practice [4]. Currently methods such as image grammars and split-and-merge expectation maximization techniques are being investigated [17, 18]. The work using an expectation maxi-

mization algorithm could potentially benefit from being seeded with initial locations of canes.

2.2 Skeletonization

Skeletons can be used to model objects which possess a network-like structure in nature. Skeletonisation involves the extraction of a skeleton from a binary digital image [11]. Skeletonisation algorithms can be split into two categories, discrete and continuous [13]. These algorithms have been used on images containing a wide variety of objects; including those containing plants and trees. Skeletonisation algorithms aim to produce skeletons which accurately represent the topology of structures present within the original image. A high quality skeleton will preserve the connectivity of the original structure, while also being both well localized and thin. These models can then be analysed and used within the application, for tasks such such as reconstructing a handwritten document in digital format, or building up a 3D model of the branches within a tree.

Previously Gittoes and Botterill [11] have analysed the application of skeletonisation algorithms in producing models of vines from a 2D image. In their research 5 discrete and 5 continuous skeletonisation methods were implemented and evaluated. The skeletons produced by all of these algorithms were compared against hand drawn ground truths. Metrics for measuring the accuracy, connectedness and thinness of the skeletons were implemented and used to quantitatively assess the skeletons. Of these algorithms they found that morphological thinning combined with an active contour model using a blurred vine image to be the most suitable skeletonisation method. This was due to the comparatively high accuracy relative to the other algorithms evaluated. An example image produced by this method can be seen in Figure 2.1.

These skeletons have not been used within the pruning project due to a number of issues. The resulting skeletons often contain errors which are caused by the presence of junk and noise within the image [4]. In addition the algorithms are not scale invariant and produce different models from the same image at different resolutions. In [6] and [12] skeletonisation methods have been assessed for their usefulness in modelling plants, with similar results. It is therefore desirable for any method developed to be capable of coping with any noise or junk which may be present within the image. In these situations it is preferable to omit the information, rather than produce best guesses which may result in errors.

2.3 Expectation maximization

In [17] a framework using expectation-maximization (EM) for segmenting 2D vine images has been proposed. This approach involves defining a probabilistic model which is used to represent segments of canes. Using this model a probability that a pixel belongs to a model can be calculated, based on the pixels distance to the model. With the models defined expectation maximization is then performed using the binary image containing the canes. In order to use the

(a) Original colour image (b) Skeleton produced with crossing-points resolved

Figure 2.1: Example input image and the resulting skeleton produced [11].

expectation maximization framework on a given input image, a set of initial models for canes has to be created. Both the parameters for the models, as well as the number of models to be initialised have to be chosen. The authors have found that this initialisation process is critical in producing a high quality segmentation of canes as a poor initialisation will result in the algorithm becoming stuck in a local maximum [17]. In this system initialisation has been done manually by the user. This approach was found to produce the best results, however this will not be feasible for use in an autonomous vine pruning robot.

In [18] this system has been further enhanced, using a split-and-merge approach as originally proposed in [27]. This allows EM to run, while altering the number of models between iterations. Splitting involves creating new models from an existing one which does not fit its data points well. Merging is performed on multiple models which bare similar properties. This step reduces the risk of the system getting stuck in a local maximum, which was one of the main issues of the first approach. In order to reduce the effects of poor initialisation random-restarts have been used. This involves randomly selecting the parameters of the initial models. This is done repeatedly and the solution which contains models which have the best fit relative to all others is selected.

The use of random restarts has shown to be more effective than heuristics previously used in [17]. Despite these improvements, additional work can be done to provide a more accurate set of initial parameters. This will likely improve the final results after running the EM method with split-and-merge included. It is hoped that the results of our research may be beneficial for the use in initialising a higher level algorithm such as this.

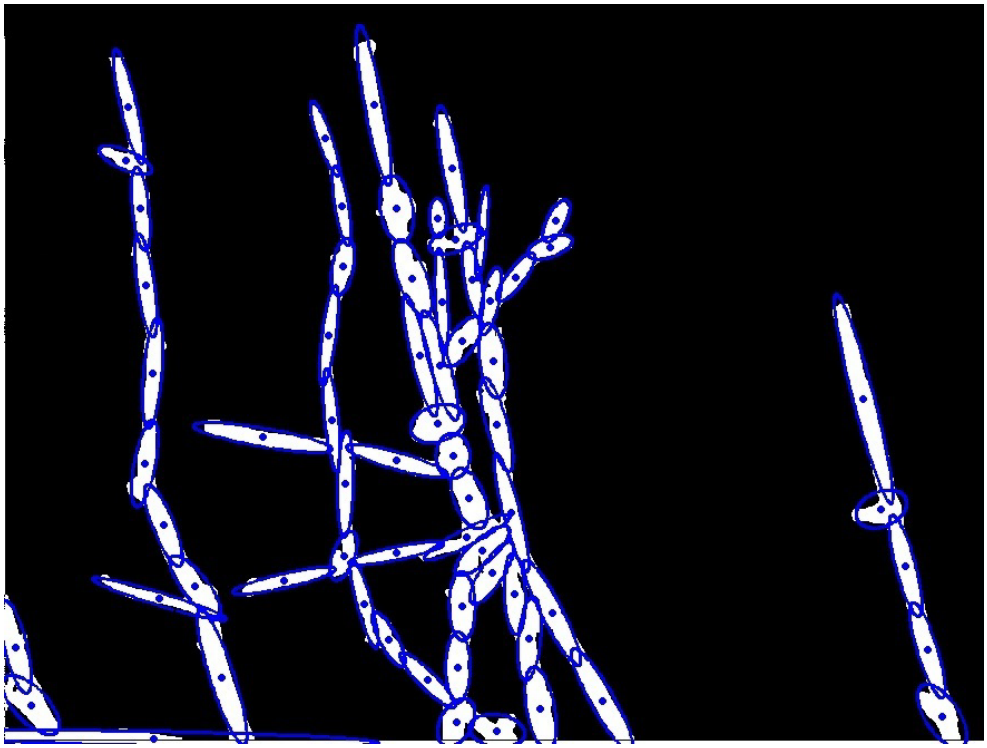


Figure 2.2: Expectation maximization convergence using a gaussian model [17].

2.4 Segmentation

Previously Flowers [10] has investigated the use of a variety of segmentation methods for use on images of vines. A selection of 5 different segmentation techniques used within related fields were evaluated on images of vines with posts, wires, canes present. Ground truth segmentation images were constructed from the images and these were then used to calculate the accuracy of segmentations produced. Mean-shift clustering was found to be the most appropriate algorithm due its high accuracy and variable number of clusters created. An accuracy of 99% was found using mean-shift on their test images.

Despite creating highly accurate segmentations with a variable amount of these segments in an image, we have not made use of mean-shift segmentation as a pre-processing step. The resulting segments produced have not been classified into vine, post or other objects within the scene. This makes it difficult to directly use this output in our research without investigating how accurately these segments correlate to actual regions within the original image. Given the proportionally high number of segments created, a mean of 6,587 in their analysis for images of 160x160 (25,600 pixels total), it seems that the image has been over-segmented. In addition to this, within their evaluation they have not made use of full resolution images, but rather smaller 160x160 sections of the original image. We will be working with full images of the plant at a resolution of 960x1280 and it is unclear on how well these methods will perform in this setup. Additionally

the current run-time performance of these algorithms is too slow even on these smaller images for use within a real time system [10]. Runtime performance is important as this segmentation would be used as a pre-processing step and therefore needs to have a minimal amount of overhead.

2.5 Plant and tree reconstruction

There has been much research in the area of plant and tree based reconstruction. This research focuses on constructing a model of the plant using a wide variety of inputs such 2D images, point clouds, laser lines and manual user input. The main goal of such works is to accurately reproduce a 3D model of the plant, including both its branch structure and foliage. The automatic recreation of the plants branch structure is of most interest to us.

In [20] Quan and Tan propose a system which can be used to automatically generate models of smaller plants. Their approach uses structure from motion to automatically generate a 3D point cloud of the plant [16]. These points are then segmented using a graph partitioning algorithm, the results of which are refined by user input. Branch reconstruction is done manually by the user using a specialised tool to assign leaves as well as create the branch hierarchy. This system has since been enhanced by Tan et al. [26]. The enhanced system is capable of reconstructing full trees, including automatically identifying portions of the branch structure and its foliage. Branch structure extraction is performed using a graph construction algorithm which assigns 3D points to branch nodes based on their distances to neighbouring points. This branch structure is then manually refined by the user in order to resolve issues such as occlusions and missing information. An example of the inputs, intermediate states and the final model produced can be seen in Figure 2.3. Despite the improvements made to their system, it still relies on manual user input in order to fully reconstruct the tree, which would not be suitable in a fully automated system such as the one we are developing.

Figure 2.3: Example of a tree being reconstructed as produced by [26]. From left to right: 1 of 18 input images, fully reconstructed branch structure, fully reconstructed tree, tree from a different perspective.

3 Implementation

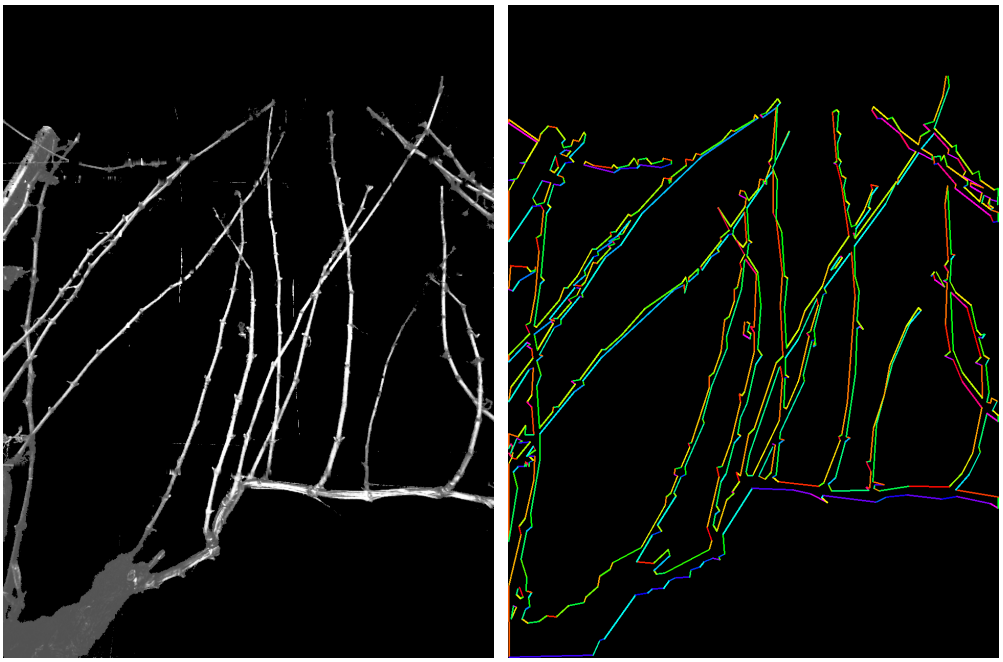
This chapter contains the details of the proposed method we have implemented for extracting the structure of the vine from a 2D image. Our approach works by extracting the edge information from a background seperated image. These edges are then iteratively joined using a set of rules. Finally models of the canes are produced by selecting edge pairs which best match a set of criteria.

3.1 Image pre-processing

The input to our system is a 2D image captured by one of the three optical cameras on the robot. Background separation is then performed, resulting in a grey scale image where the intensity of each pixel is the probability that it is in the foreground. Background separation is performed using Bayes formula. In addition to this, all wire pixels have been removed from the image using the current wire detector. Both of these methods were previously developed by Botterill and Green [3] and their details are beyond the scope of this report. Finally the image is converted to a binary image, where all pixels with a foreground probability larger than 0.5 are set to 1, enabling us to perform contour extraction. An example background segmented image with wires removed can be seen in Figure 3.1a.

3.2 Contour Extraction and Approximation

Using the pre-processed, background separated image we can extract the topological information of the vines by performing contour extraction. This provides us with information about regions within the image, including their hierarchical relationship with one another. This information is the basis for our vine modelling method. Contour information has been extracted using the method described in [24], the implementation of which was provided by the OpenCV library [5]. This technique builds upon existing border following algorithms [23], while also constructing a full hierarchy for all identified regions. This provides us with all pixels that form each border as well as their corresponding parent region and siblings.



(a) Background separated image. (b) Overlaid pruned edge pairs.

Figure 3.1: Greyscale image after performing background subtraction, along with the extracted contours which have been approximated into line segments. Line segment colours are based off of their direction.

After performing contour extraction we make use of the Ramer-Douglas-Peucker [21, 8] algorithm in order to generate line segments which consist of a minimal number of points, without significant variation from the original contour. This step is performed in order to reduce the amount of data and move to a higher level of abstraction. By forming line segments from these pixels we can obtain a richer representation of the contour. This enables us to make decisions based on the length and direction of the line segments. The Ramer-Douglas-Peucker algorithm achieves this by recursively simplifying the set of points into lines. The start and end point for the original contour are selected as points to be kept, and the furthest point between these is found. If the distance between the line segment and this point is greater than the error threshold, this point is kept, and the two new resulting line segments are subdivided. This continues until we are left with a simplified representation of the original set of points.

From the approximated line segments we are also able to calculate the direction that the segment is oriented within a range of $0-360^{\circ}$. Having the orientation expressed within this range enables us to determine whether segments are running parallel, and whether they are oriented in opposing directions. This is necessary, as it aids in the identification of opposing cane edges, which are oriented in such a way that they will run in the opposite direction.

3.3 Association criteria

The edge association portion of the method plays a critical role in the overall process of building a model of the cane segment. This step consists of a series of processes which attempt to identify single, unoccluded edges along a single cane segment. During this stage only line segments and their neighbours are considered. Line segments consist of two ordered points, representing the pixel locations on the image. Using these lists of segments, the relationships of the segments with their neighbours are used to build up associations between segments. These built up associations can then be used at a later stage when creating edge pairs and modelling the cane segment.

3.3.1 Associations vs. Simplification

In order to build a model of candidate edges for each cane, two approaches were implemented. The first implementation involved iteratively simplifying sets of line segments, while our second approach built up associated segment sets.

Simplification involved merging all connected line segments involved into a new, single line segment. For this method both the straight line detector and bud detector were implemented. If two or more segments were identified as belonging to a bud or straight edge, all segments involved were replaced by a single new line segment which contained the start and end points for the first and last segments involved, respectively. This method was found to be inadequate, as it rapidly diverged from the underlying model and this resulted in many poor decisions as it

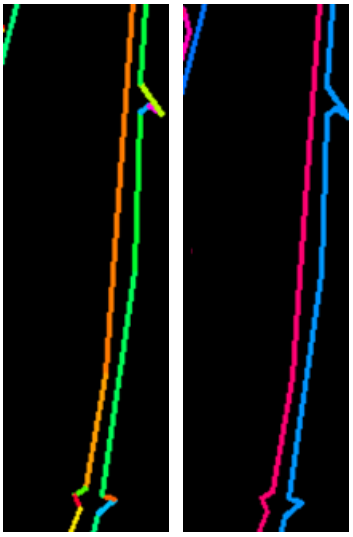


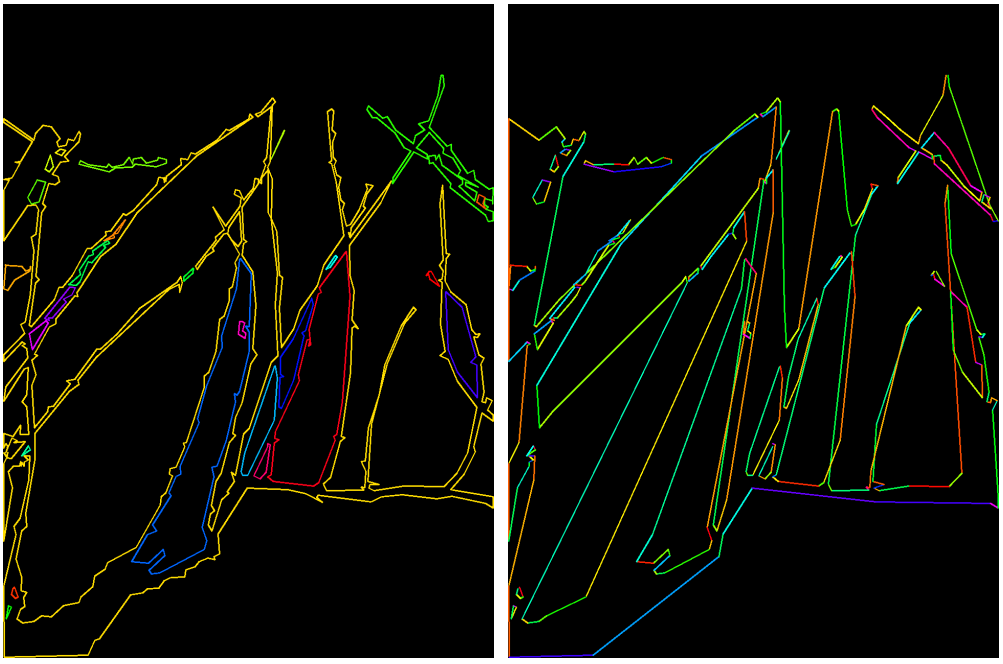
Figure 3.2: Before and after performing edge association. All buds and growths have been detected successfully, as denoted by the colours of the line segments.

progressed. An example of the final result after performing segment simplification can be seen in Figure 3.3

Due the limitations caused by iteratively simplifying line segments we implemented a second approach using associations only. This approach involved building up associations between individual line segments without changing or removing any of the original segments. After the approximation algorithm has generated line segments, each segment is placed into its own set. As features are detected between individual segments, the union of all segments and their associations is calculated and assigned to each segment involved. This enables us to distinctly determine which segments are associated with each other, allows us to build up single edges of a cane. There are a number of advantages of this approach over iteratively simplifying line segments. Since the underlying segments are never altered, any association methods which relies solely on the attributes of the segments, such as their length and direction need only to make a single pass over the list of line segments, thereby reducing the number of operations. Association methods which rely on other properties, such as those of the association sets themselves however will still have to be iteratively run until no features are found.

3.3.2 Near Straight Lines

The first and most simple association method aims to identify connected line segments which have a similar direction within a specified threshold. Every segment within the contour is iterated over, and the difference in direction between the current and the next connected segment is calculated. If this difference is minimal, associate them together. By associating segments which have approximately the same direction we are able to build up much of a single edge along the



(a) Original extracted contours. (b) Resulting simplified segments.

Figure 3.3: The result of performing segment simplification. The final image does not resemble the original set of contours very well, which makes identifying canes very difficult.

cane, generally only stopping at points of occlusion or large growths. This is the most simple detector and accounts for the majority of associations built within the edge association phase. Since we are not altering any of the underlying information when building associations, this step can be performed in a single pass for each contour extracted. It is worth noting that the segments that belong to the contour form a cycle. We therefore only check the next connected segment to avoid redundant operations.

3.3.3 Buds and Tendrils

Canes often have buds, tendrils or other growths coming off of them. These growths are generally small and dont alter the shape of the cane significantly. In order to reliably detect the full edge of the cane these growths need to be considered when associating segments. To accomplish this a bud detector has been developed which is capable of identifying a variety of buds and growths. The outline of our implementation can be seen in Algorithm 1.

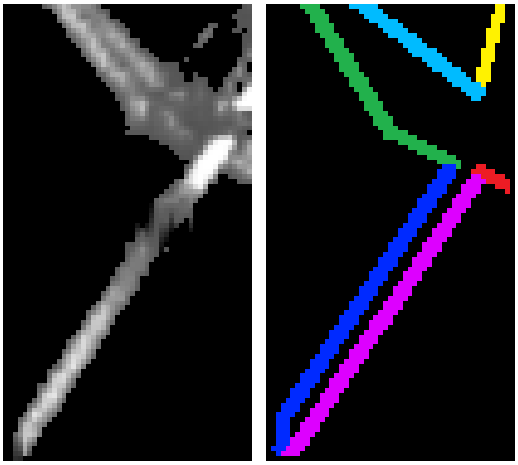
The algorithm attempts to identify segments which lie within a short distance of each other and run along in an approximately similar distance. In order to accomplish this, the algorithm repeatedly iterates over all segments within the contour and selects the Nth connected segmented ahead of the current. The value of N is incremented between iterations up until a predefined

Algorithm 1: Bud detection

```
for i = 1 to N do
    foreach contour in image do
        foreach segment in contour do
            comparisonSegment = currentSegment.skip(i);
            distance = 0;
            foreach segment in-between current and comparison do
                distance += segment.length;
            end foreach
            directionDifference = abs(currentSegment.direction -
            comparisonSegment.direction);
            if directionDifference < DirectionThreshold and totalDistance <
            MaxDistance then
                Associate all segments in-between current and comparison;
            end if
        end foreach
    end foreach
end for
```

maximum. Changing N is necessary as buds may consist of as few as 2 or as many as 10 or more segments. After selecting this segment the cumulative length of all segments between the current segment and the selected segment is calculated. If this distance is small, and the difference in heading between the current and selected segment is minimal, we can associate the current segment, the selected segment, and all segments in-between. This can be seen more clearly in Figure 3.4.

Figure 3.4: Bud detection with N = 3. The two red segments have their direction compared and the total length of the green segments is calculated. Since Both red segments run in the same direction, and the cumulative distance covered by the green segments is minimal, all segments involved will become associated.



(a) Background subtracted segment

(b) Resulting edge associations

Figure 3.5: Without the length constraint, the feature detector would incorrectly classify the pink and blue edges as a bud.

This method of detecting buds and other growths relies on the fact that buds generally form a set of small connected segments which shortly return back to the main cane and then continue approximately in the original direction. We limit the cumulative length of all intermediate segments in order to prevent false positive detections due to occlusions of other branches and objects. An example of this can be seen in Figure 3.5, where a straight branch of another cane could have been falsely identified without the length constraint. Since this feature detector does not alter the underlying segments being compared, it is performed in a fixed number of passes.

3.3.4 Weighted directions

Our final edge association method works on the entire segment association sets, rather than individual segments and their local neighbourhood. This algorithm attempts to associate entire sets based on their overall average weighted direction of the individual segments comprising the set. This is necessary as the bud detector can fail on cases where the first and last segment forming the bud do not quite align with respect to their direction, despite the overall structure of the segments running approximately in the same direction. By calculating the weighted average distance for the entire set we can reduce the impact short bud segments have on the overall decision making process. This is desirable as buds tend to run in directions which do not coincide with the overall direction that the cane is growing in and therefore should have less weight on the decision.

Our implementation works by iterating over all segments within each contour of the image. For each segment, the current segment and the following segment are considered. If these seg-

Algorithm 2: Weighted directions

```
foreach contour in image do
    while Associations made or first iteration do
        foreach segment in contour do
            nextsegment = currentSegment.next();
            if currentSegment is not associated with nextSegment then
                if difference in weighted direction between current and next is acceptable
                then
                    associate the two segments and their sets;
                end if
            end if
        end foreach
    end while
end foreach
```

ments are not currently associated, we calculate the weighted average direction for both of these segments and all the other segments associated with them. If the difference between the two weighted directions for both segment sets is within a threshold, we merge the two sets together. This is done repetitively until no sets with similar weighted average directions can be merged. Unlike the straight line and bud detectors this method makes a variable number of passes over the set. This is due to the fact that the underlying information, the overall weighted direction of segment sets changes whenever a merge is performed.

3.4 Edge pairing

After all the feature detectors have executed most individual line segments will have been merged into distinct sets forming their associations. These sets generally form single edges along the side of the cane up until points of occlusion. It is now necessary to start building up models of the cane section using these segment sets. The first step in this process involves attempting to identify edge pairs which match the following criteria.

1. Pairs have a minimal distance apart
2. Run in opposite directions, being roughly parallel.
3. Do not match holes in the original image.

An ideal edge-pair will have a very minimal overall distance between the two opposing edges, run in opposing directions and not match a hole. Distance should be minimised as canes tend to

have a relatively consistent thickness without significant variation. During the contour extraction and approximation stage, contours are extracted in a clockwise order. This gives us the guarantee that any opposing edge will be oriented in the opposing direction. Finally when pairing edges, we do not want to match holes. Holes are defined as areas where there is no cane in-between the two edges.

The following method has been used to match edge pairs together. Every segment set is iterated over, and compared with all other sets. Set comparisons are performed by iterating over each segment within the set. For each segment, we find the closest point in the candidate set relative to the start and end point for the current segment. Using this we can calculate the distance from that point as well as the difference in direction. This enables us to calculate an average distance and difference in overall direction between the two sets, enabling us to determine how well the pair match. This process can be seen in Figure 3.7a.

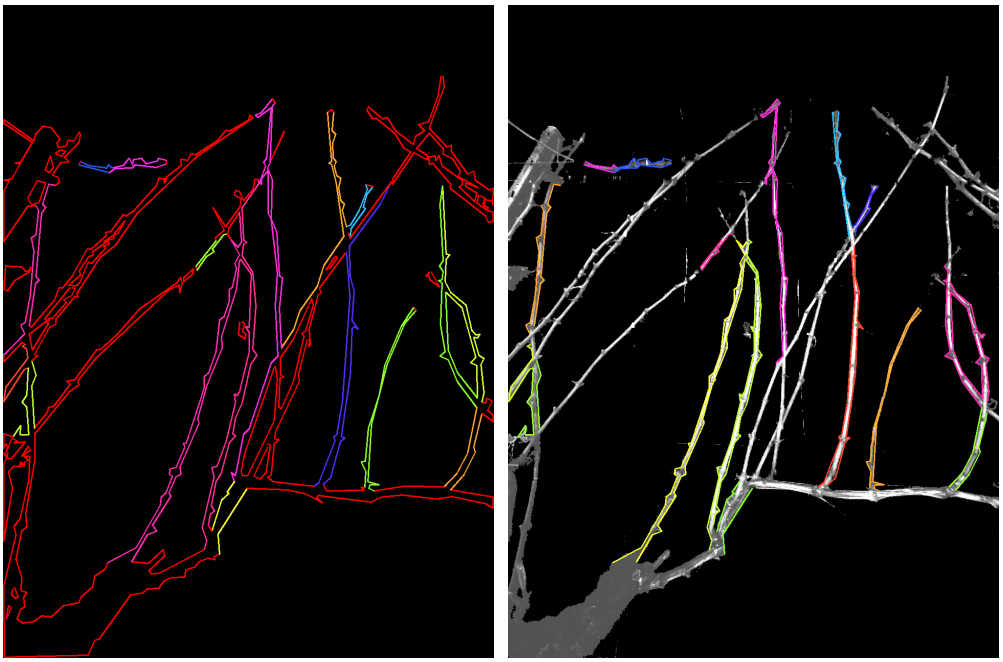
After comparing all segment sets with one another we can then select pairs which match. In order for a match to be made, the two segments must have an average distance less than a certain threshold and their difference in direction must be approximately 180°. Finally both segment sets must agree on each other being the best candidate. This is necessary in order to prevent any segment sets from being used in multiple models.

The final criteria when creating edge pairs is to not match holes. Holes are contours which surround a blank area within the image, an area in which there are no foreground pixels. Holes can often be erroneously identified as canes when there are two or more canes running parallel to one another. When this occurs, it is ambiguous as to which edges should be paired together, as they will all tend to be running in opposing with similar distances between one another. In order to prevent this from occurring the contour hierarchy is used to determine whether two segment sets are likely to surround a hole. By using the hierarchy we will not match any two segment sets which are both interior contours, as this guarantees that the area between them is a hole. This prevents the majority of holes being selected as edge pairs without any need to refer back to the original image.

3.5 Cane modelling

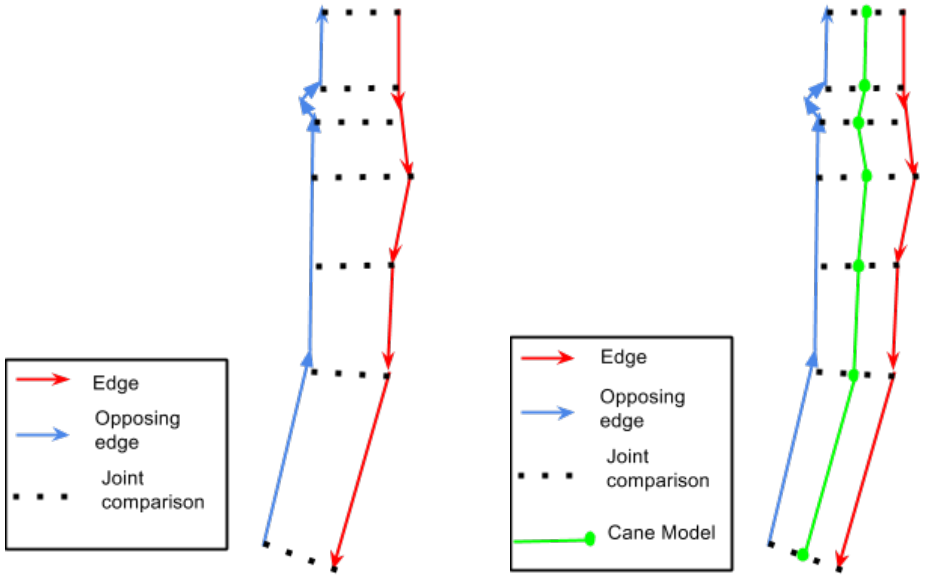
Using the edge pairs selected in the previous step it is now possible to generate models of the cane. Prior to generating the model we have found it necessary to partially remove segments from either pair that are too far from the opposing side. This is done by finding the closest point on the opposing edge for every segment. If the closest point is beyond a threshold distance, we remove this segment from the pair. This is necessary as edges of the cane may have sprouts or other growths coming off of it, which results in segments beyond the growth not being associated together. This step can be seen in Figure 3.6.

Creation of the model is performed in a manner baring much resemblance to the edge pairing



(a) Resulting edge pairs. (b) Overlaid pruned edge pairs.

Figure 3.6: Edge pairs before and after pruning off segments which do not correspond well with the opposing edge. Red segments in the first image denote segments which have not been paired up.



(a) Edge pairing procedure. (b) Model creation.

Figure 3.7: Edge pairs are compared using their joints and their average difference in direction. A model of these pairs is then created using the midpoints between segment joints.

procedure. For each pair of segment sets, we iterate over all segments in both pairs. For every segment we calculate the closest point on the opposing segment set for both the start and end point of the segment. The midpoint between these is calculated and this is used as a point in the generated model. The distance can also be stored in order to retain width information about the cane at every control point. All points are then sorted along the Y axis, and this forms the model of the cane. Figure 3.7b visually demonstrates this procedure.

4 Results

This chapter contains the quantitative results of our evaluation of the method described in Chapter 3. We have selected two methods for assessing the cane models produced, both of which rely on comparison against ground truth models. These methods provide insight into how well the developed method performs, with a focus on either overall accuracy or error rate.

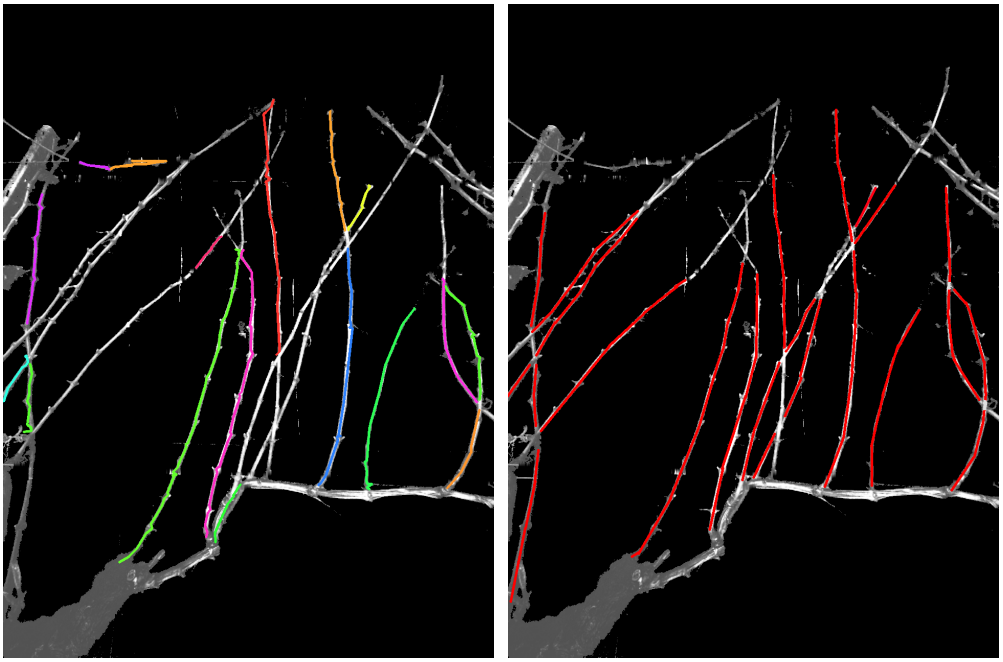
In order to determine how effective our method for identifying canes is, we need to a way to quantitatively compare its output with the ideal models. Since there are no methods for reliably and accurately identifying canes automatically we will be comparing our results with hand drawn ground truth models. To accomplish this a set of 8 distinct images of vines from our robot have been selected. This set consists of a variety of images, including different plant species as well as significantly different perspectives of the same plant. For all of these images we have created a separate ground truth version which has the centre of the cane marked out in red. When creating these ground truth images we have marked out pixels which we believe best correspond to the centre point of the cane. Areas in which it is ambiguous as to whether a cane is present or at points of occlusion have been omitted in our ground truths. An example of our generated models and the corresponding ground truth can be seen in Figure 4.1. Using these ground truth images we can calculate the level of correspondence our generated models have.

4.1 Overall accuracy

Previously Gittoes et al. [11] have measured the accuracy, connectivity and thickness of skeletons that were generated in order to model vines. Of these metrics, only accuracy can be applied to the models our algorithm produces. Connectivity is not an issue in our generated models, as they are all inherently connected up until points of occlusion, which we do not resolve. As a result we have not included this metric within our analysis.

Thickness was previously used to measure the mean width of the generated skeleton, where the ideal thickness was 1 pixel wide. Our method generates models which are a single pixel wide; therefore including this metric into our analysis would be redundant.

The method used in [11] to measure accuracy is an adaptation of earlier work in quantitatively assessing the quality of skeletons [14, 15]. Their method involves calculating the mean distance



(a) Generated models. (b) Hand drawn ground truth.

Figure 4.1: An example of our generated models alongside the corresponding hand drawn ground truth.

between pixels on the generated image and the hand made ground truth. In [11] this has been slightly altered and instead is expressed as a percentage rather than distance.

$$S_{g,t} = \frac{1}{2}\left(\frac{M_{g,t}}{N_g} + \frac{M_{t,g}}{N_t}\right) \tag{4.1}$$

Equation 4.1 calculates the overall accuracy of a generated model, g and the corresponding truth, t. In the equation $M_{g,t}$ is the number of pixels in the generated model that correspond to a point on the ground truth. N_g is the number of pixels in all the ground truth models, while $M_{g,t}$ is the number of pixels on the ground truth that have a corresponding pixel on the generated model, and N_t is again the total number of pixels in the generated models. The result of this, $S_{g,t}$ is the similarity between the generated models and the ground truth.

In order to determine whether a pixel in the ground truth or generated models has a corresponding point, we find the nearest pixel and calculate the distance between them. If this distance is less than a threshold value, we mark the pixel as found. This step is repeated for all pixels in the ground truth and generated models. In [11] a value of 4 pixels was used as the threshold distance. Since we are working with similar images of approximately the same resolution we have used this value in our analysis as well.

4.2 Error rate

In addition to the overall accuracy measure, we have used an additional metric to gauge the correspondence of our generated models with their ground truth. This equates to only using the error rate portion of Equation 4.1. Instead of calculating the correspondence of the generated model to the truth as well as the truth back to skeleton, we only consider how much of our generated model directly corresponds to points on the truth. This metric is of relevance as we have focused on reducing errors in our resulting models, rather than producing more models at the cost of increased errors.

$$S_{c,t} = \frac{M_{t,g}}{N_t} \tag{4.2}$$

Once again the resulting value will be in the range of 0 to 1, where a value of 1 means that everything generated is completely error free. It is worth noting that this metric is biased towards generating as few models as possible, however we are not using it to tune the parameters used in the algorithm.

4.3 Performance

Although run-time performance has not been a focus of this research, we have measured the time it takes our implementation to identify canes. Results have been collected on a system equipped with an Intel i5 3470 CPU, 8GB of memory running Windows 8. The results of these timings can be found in Table 4.1.

4.4 Results

We have used a set of 8 images of various grape vine plants taken from the robot. Using these images overall accuracy, ground truth correspondence and run-time performance have been measured. The results for each of the images tested can be seen in Table 4.1. From this we have calculated an overall average accuracy of 0.713 with a standard deviation of 0.059. Average correspondence with ground truth was found to be 0.908 with a standard deviation of 0.0414. Finally average time taken per frame was calculated to be 47.93ms with a standard deviation of 12.12ms.

Image No	Overall accuracy	Ground truth correspondence	Time taken (milliseconds)
1	0.700	0.904	55.81
2	0.645	0.860	25.92
3	0.621	0.825	42.27
4	0.700	0.922	47.45
5	0.690	0.956	63.85
6	0.785	0.944	52.35
7	0.774	0.923	60.71
8	0.789	0.931	35.13
Mean	0.713	0.908	47.93
Standard deviation	0.059	0.0414	12.14

Table 4.1: Overall accuracy, correspondence with ground truth and time taken for all images we have used.

5 Discussion

This chapter contains an overview of the results found in Chapter 4. Factors such as overall accuracy, correspondence of models with ground truth and run-time performance are analysed and discussed. We provide insight into why the method has performed as it has and briefly mention possible improvements which could be made to address any shortcomings. Finally a comparison with some of the skeletonisation methods assessed in [11] is made.

5.1 Accuracy

We have measured the overall accuracy of our method using the same process defined in [11]. From this we have found a mean accuracy of 71.3% with a standard deviation of 5.9%. These results are promising, however it is unlikely they are high enough for use in reliably reconstructing the 3D model of the canes. There are a number of issues and limitations with the models found which directly impact our accuracy values. The main reason for the low overall accuracy is the low number of models generated compared to the number identified in the ground truth. This is supported by our results for the correspondence with the ground truth. When comparing the models our method has generated against the ground truth only, we have found a mean accuracy of 90.8% with a standard deviation of 4.1%. This is significantly higher than the overall accuracy and supports the claim that the main reason our accuracy is low is due to missing information. The bulk of the models created do correspond to points within the ground truth. When manually comparing the models produced with the ground truth, in some cases our method has identified cane portions which are not marked in the ground truth, but could potentially be canes. This can be seen in Figure 5.1. When creating the ground truth images, portions of cane which appeared ambiguous were omitted.

(a) Possible piece of cane. (b) Identified model.

Figure 5.1: A model which does not correspond to anything in the ground truth, thus resulting in a reduction in measured accuracy.

This result is promising as it means that the majority of the models identified are correct. It may therefore be possible to make use of the models we have identified as initial locations for a higher level algorithm such as the one in [18]. Additionally there are a number of improvements which can be made to reduce the missing information in our models. Some of the main improvements which could be investigated are outlined in Chapter 6.

5.2 Skeletonization comparison

In [11] several skeletonisation methods have been quantitatively assessed for their use in extracting the structure of vines. Although our approach does not produce a skeleton, we have extracted a structure which can be compared with a skeleton. As a result we have made use of the same accuracy metrics in our research. This has enabled us to compare our results with prior research, allowing us to gauge the relative performance of our method. Figure 5.2 contains the overall accuracy of the skeletonisation algorithms assessed in [11] alongside our own. Error bars are for a 95% confidence interval.

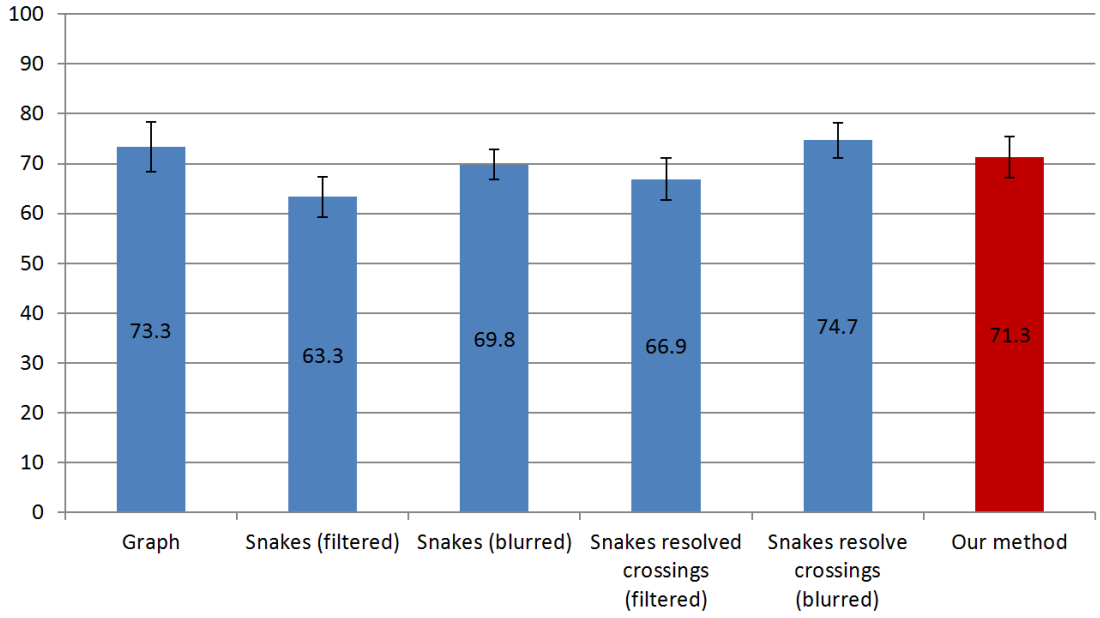


Figure 5.2: Overall accuracy of our method compared with skeletonization methods on vines as conducted in [11]

From the figure we can see that our method had a lower mean overall accuracy when compared to both the graph building and the snakes with crossing point resolution using the blurred image skeletonisation methods, while scoring higher than the rest of the skeletonisation methods. Despite this there is insufficient evidence to suggest that there is a significant difference between these methods using a 95% confidence interval. It is likely that with the changes proposed in Chapter 6 that our approach will surpass the usefulness of these skeletonisation methods when

modelling vines.

5.3 Performance

Currently the three cameras on the robot capture frames at a rate of 7.5 per second [3]. Therefore in order for the system to run in real-time as the robot moves along the plant features within an individual frame should be extracted within a period of 133ms, assuming that the three frames can be processed concurrently. Using our set of 8 images, we found the mean runtime to be 47.93ms, with a standard deviation of 12.14ms. This enables our system to process images at a rate of 20.8 frames per second on average, which is above the minimum rate of 7.5 that the system currently records at. In addition our method operates independently of other frames and therefore could be applied in parallel for each of the cameras equipped on the robot. If our method is used as pre-processing step for an algorithm such as the one currently being developed in [18], it would be a necessity for the runtime of it to be minimal.

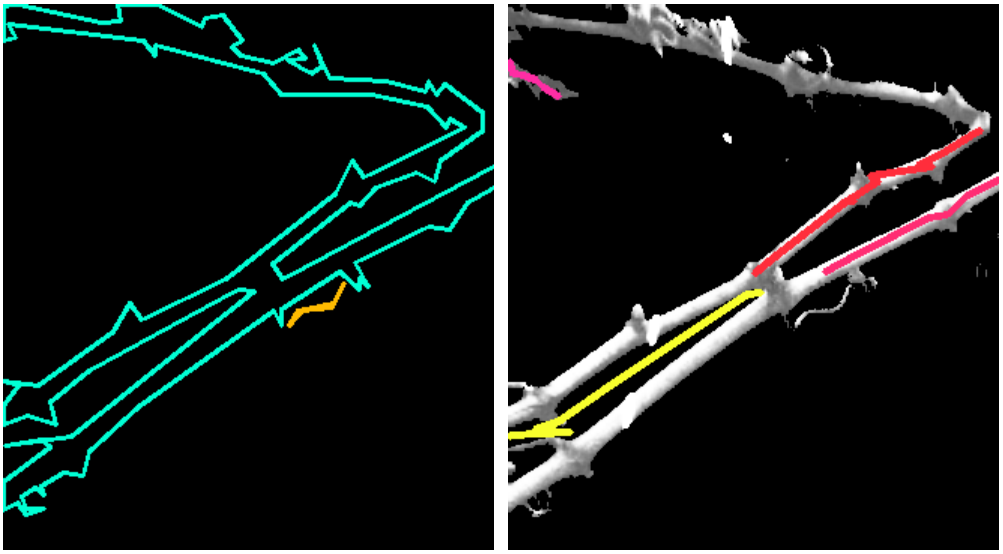
6 Limitations and Future Work

In this report we have described our implementation of an algorithm which makes use of contour information in order to construct partial models of canes using the topological information extracted from these contours exclusively. This method has been shown to be capable of extracting much of the canes structure; however there are a number of improvements which could be made in the future in order to further increase the accuracy of the algorithm. Some of these improvements include changing the hole detection method, automatically tuning parameters and resolving occlusions.

6.1 Hole Detection

Holes are regions in the image which surround black, empty pixels which have been classified as the background. In our system these are regions which are fully surrounded by canes and other objects within the image. When matching edge pairs in order to model the cane it is important to not match two pairs which surround a hole. Our current implementation attempts to avoid this by making use of the hierarchy provided when extracting contours. This successfully eliminates the majority of holes being matched, however it is still possible for two edges surrounding an empty region to be incorrectly paired up. Figure 6.1 demonstrates this. In this example the algorithm has matched two edges which surround a blank area within the image, but are not a hole as defined by the hierarchy.

This is because both of these edges belong to the same contour which has no parent, making it a viable match. In situations such as these it is not possible to determine if the matched pair surrounds a hole in the image using the contour hierarchy alone. In order to prevent this from occurring additional checks should be made. The most intuitively obvious approach would be to make use of texture information available in the original image. If the control points of the modelled cane produced by the edge pair results in the majority of points falling on pixels which are black, we have likely matched two edges surrounding a hole. Any methods for improving hole detection will result in a reduction in the error rate of the algorithm, therefore further exploration into this will be valuable.



(a) Extracted contours (b) Resulting models

Figure 6.1: An example of the current method of hole detection failing. The yellow cane model falls between the area inbetween two different canes.

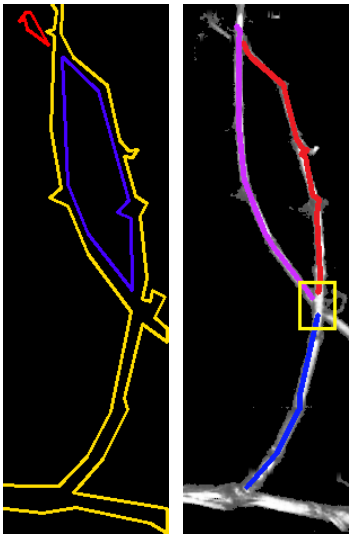
6.2 Parameter selection

Our current implementation relies on a number of parameters to be selected in order for the algorithm to work as expected. These parameters are mostly related to distance and angle thresholds used when associating edges, pruning segments and selecting edge pairs. For this implementation we have manually selected the parameters that appeared to work best for the images we tested against, using the overall accuracy as a benchmark. An alternative approach to this would be to automatically tune these parameters using a separate cost function. This would enable the algorithm to be rapidly adopted for use in a different setup with varying image resolutions and other properties.

Within the current implementation all distances are measured using pixel distances. This means that the choice of parameters for distance values is directly tied to the resolution of the image and the distance from the camera to the vines. When attempting to pair up edges, we make use of the average distance between all joints along the two edges. An alternative to this could make use of the standard deviation instead, making this portion of the algorithm scale invariant.

6.3 Occlusion resolution

In this report we have outlined a method for detecting cane regions and building models of them through the use of contours and the resulting line segments which form the edges of the



(a) Extracted contours (b) Resulting models

Figure 6.2: Cane segments have been identified succesfully up to points of occlusion, which have not been resolved.

cane. When building up the edge information for the side of the cane, only segments which are directly connected and within the same contour are considered while performing associations. This introduces a limitation in that the algorithm is unable to fully reconstruct canes in which parts of them are occluded by other objects within the image. This is because occlusions often result in the contour extending around the borders of the occluded region, as show in Figure 6.2a. This makes it impossible for our method to resolve occlusions, resulting in separate partial cane models for regions between occlusions. All of the images in our data-set have had at least a single pair of vines which have occluded one another. It is therefore desirable for the algorithm to be able to detect and resolve these occlusions when they occur.

One approach which could be investigated would involve iteratively joining existing models which have been produced by our current method. By comparing both the straight line distance and the difference in average weighted directions of the models, it may be possible to reliably join models which have occlusions between them. This approach would work specifically in instances where the algorithm has been able to reliably produce full models between points of occlusions, such as those highlighted in yellow in Figure 6.2b.

Additionally improvements could be made to the edge association portion of the algorithm. After performing all straight line and bud associations, a new cross contour association method could be applied. This step would likely involve individually comparing two straight line segments, scoring their likelihood of belonging to a single cane based on their distance apart, as well as factoring in the difference in average weighted direction of their segment sets and any other relevant properties. This would enable the algorithm to determine where the cane edge exists

in regions where it may be occluded. In addition it would also enable full reconstruction of canes without any changes being made to the current edge pairing and model reconstruction implementation.

6.4 Performance

As mentioned in Chapter 4, we have no placed an emphasis on runtime performance of our method during implementation. As a result there is much which can be improved in our implementation in order to drastically improve its performance. Future research could involve thoroughly exploring the runtime performance of our method. This would involve rigidly measuring the performance and determining what factors negatively affect it. Additionally methods for reducing the run time cost of our method could be explored. Spatial partioning data structures such as Quad trees [9] or k-d trees [1] might prove useful in this regard. Using a spatial data structure will remove otherwise unnecessary comparisons in several steps within the algorithm. For example when attempting to match edge pairs, those pairs in which all segments lie beyond a threshold distance need not ever be compared. By storing all edge-pairs within this structure, we would be able to efficiently query only those which could feasibly be a valid pair. This would significantly reduce the number of comparisons performed in this step alone.

7 Conclusions

Within this report we have proposed a new method for extracting the structure of vines from 2D images. A wide variety of methods have been used in the field of plant modelling and reconstruction [20, 26, 19, 22], however very few have focused on accurate, automatic reconstruction of the plants branch structure. More recently work has been undertaken to investigate the usefulness of skeletonisation algorithms to model branches in fruit trees [6] and vines [11]. These skeletonisation methods have been successful in extracting some of the structure, however it has not been adequate for building up a 3D model of the plants.

Our method uses edge information obtained by using a border extraction algorithm [24] on the background subtracted image. Using a set of simple rules edges are iteratively joined up based on their properties. Finally edge pairs are compared and those which possess properties of canes within the image are transformed into models. This method has been quantitatively assessed against hand drawn ground truth models which are near perfect. From this we have found a mean overall accuracy of 71.3% ($\sigma = 5.9\%$), with 90.8% ($\sigma = 4.1\%$) of the generated models matching the ground truth.

These results are promising, but are unlikely to be adequate to be used directly within the reconstruction of the 3D model of the plant. Instead additional improvements which are likely to increase the number of models detected have been proposed. The majority of these improvements involve increasing the number of models that are extracted, while retaining high accuracy and low error rates. Resolving points of occlusion, preventing hole matches and improving the edge association process are all areas which could be explored to further enhance our method.

Alternatively due to the high accuracy of the models our method does identify, as well as the relatively cheap run-time cost, it is likely that our method could be used to initialise a higher level algorithm such as expectation maximization [18]. The split-and-merge expectation maximization algorithm proposed by Marin [18] has shown promising results and could potentially benefit from the results of this research.

Bibliography

[1] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[2] Tom Botterill, Steven Mills, and Richard Green. Design and calibration of a hybrid computer vision and structured light 3d imaging system. In *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*, pages 441–446. IEEE, 2011.

[3] Tom Botterill, Richard Green, and Steven Mills. Reconstructing partially visible models using stereo vision, structured light, and the g2o framework. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*, pages 370–375. ACM, 2012.

[4] Tom Botterill, Richard Green, and Steven Mills. Finding a vines structure by bottomup parsing of cane edges. *In Proceedings of Image and Vision Computing New Zealand*, 2013.

[5] G. Bradski. Opencv, March 2013. URL `http://opencv.org/`.

[6] Alexander Bucksch and Stefan Fleck. Automated detection of branch dimensions in woody skeletons of leafless fruit tree canopies. In *SILVILASER*, 2009.

[7] Sam Corbett-Davies, Tom Botterill, Richard Green, and Valerie Saxton. An expert system for automatically pruning vines. In *Proceedings of the 27th Conference on Image and Vision Computing New Zealand*, pages 55–60. ACM, 2012.

[8] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[9] Raphael A. Finkel and Jon Louis Bentley. Quad trees a data structure for retrieval on composite keys. *Acta informatica*, 4(1):1–9, 1974.

[10] Simon Flowers and Richard Green. *Low-level Image Segmentation for a Vine Imaging Robot*. Honour's report, Department of Computer Science and Software Engineering, University of Canterbury, 2012.

[11] Will Gittoes, Tom Botterill, and Richard Green. Quantitative analysis of skeletonisation algorithms for modelling of branches. *In Proceedings of Image and Vision Computing New Zealand*, 2011.

[12] Hui Huang, Shihao Wu, Daniel Cohen-Or, Minglun Gong, Hao Zhang, Guiqing Li, and Baoquan Chen. L1-medial skeleton of point cloud.

[13] J Komala Lakshmi and M Punithavalli. A survey on skeletons in digital image processing. In *Digital Image Processing, 2009 International Conference on*, pages 260–269. IEEE, 2009.

[14] Louisa Lam and Ching Y Suen. Automatic evaluation of skeleton shapes. In *Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, pages 342–345. IEEE, 1992.

[15] Louisa Lam, Seong-Whan Lee, and Ching Y Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 14(9): 869–885, 1992.

[16] Maxime Lhuillier and Long Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27 (3):418–433, 2005.

[17] Ricardo Marin, Tom Botterill, and Richard Green. *Segmentation and Hierarchical Structure Retrieval of Vine Canes From 2D Images*. Phd proposal, Department of Computer Science and Software Engineering, University of Canterbury, 2012.

[18] Ricardo Marin, Tom Botterill, and Richard Green. Split-and-merge em for vine image segmentation. *In Proceedings of Image and Vision Computing New Zealand*, 2013.

[19] Boris Neubert, Thomas Franken, and Oliver Deussen. Approximate image-based tree-modeling using particle flows. In *ACM Transactions on Graphics (TOG)*, volume 26, page 88. ACM, 2007.

[20] Long Quan, Ping Tan, Gang Zeng, Lu Yuan, Jingdong Wang, and Sing Bing Kang. Image-based plant modeling. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 599–604. ACM, 2006.

[21] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972.

[22] Alex Reche-Martinez, Ignacio Martin, and George Drettakis. Volumetric reconstruction and interactive rendering of trees from photographs. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 720–727. ACM, 2004.

[23] Azriel Rosenfeld and Avinash C Kak. *Digital picture processing*, volume 1. Elsevier, 1982.

[24] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[25] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2011.

[26] Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, and Long Quan. Image-based tree modeling. *ACM Transactions on Graphics (TOG)*, 26(3):87, 2007.

[27] Naonori Ueda, Ryohei Nakano, Zoubin Ghahramani, and Geoffrey E Hinton. Split and merge em algorithm for improving gaussian mixture density estimates. *Journal of VLSI signal processing systems for signal, image and video technology*, 26(1-2):133–140, 2000.